Vec2Doc: Transforming Dense Vectors into Sparse Representations for Efficient Information Retrieval

Fabio Carrara[✉1[0000−0001−5014−5089]], Claudio Gennaro[1[0000−0002−3715−149X]], Lucia Vadicamo[1[0000−0001−7182−7038]], and Giuseppe Amato[1[0000−0003−0171−4315]]

ISTI CNR, Pisa, Italy
{fabio.carrara, claudio.gennaro, lucia.vadicamo, giuseppe.amato}@isti.cnr.it

Abstract. The rapid development of deep learning and artificial intelligence has transformed our approach to solving scientific problems across various domains, including computer vision, natural language processing, and automatic content generation. Information retrieval (IR) has also experienced significant advancements, with natural language understanding and multimodal content analysis enabling accurate information retrieval. However, the widespread adoption of neural networks has also influenced the focus of IR problem-solving, which nowadays predominantly relies on evaluating the similarity of dense vectors derived from the latent spaces of deep neural networks. Nevertheless, the challenges of conducting similarity searches on large-scale databases with billions of vectors persist. Traditional IR approaches use inverted indices and vector space models, which work well with sparse vectors. In this paper, we propose Vec2Doc, a novel method that converts dense vectors into sparse integer vectors, allowing for the use of inverted indices. Preliminary experimental evaluation shows a promising solution for large-scale vector-based IR problems.

Keywords: Inverted Index · Approximate Search · High-Dimensional Indexing · Very Large Databases · Surrogate Text Representation

1 Introduction

Deep learning and artificial intelligence have significantly changed the scientific research landscape, impacting a wide range of fields such as computer vision, natural language processing, and data science. One field that has experienced a profound transformation is Information Retrieval (IR), where the ability to retrieve information accurately and across different modalities has greatly improved. This has been made possible by the advent of neural networks, which have simplified IR problems into searches for vector similarities in latent spaces generated by deep neural networks. These advances have opened up new possibilities for the development of more sophisticated and accurate search systems.

When it comes to indexing billions or trillions of data descriptors, there is limited consensus on which algorithms are the most effective at this scale versus their hardware cost [14]. However, when dealing with large-scale databases containing billions of data objects, such as those found on the web, common approaches of comparing query feature vectors with data feature vectors can become computationally infeasible.

In the context of textual documents, full-text search engines, like Elasticsearch, Solr, or Lucene, have been extensively used and tested in various domains and have already proven to be effective for large-scale indexing and searching. They leverage mature technology and infrastructure, offering scalability, flexibility in querying, and a rich set of features. For these reasons, over the years, we have explored various approaches to enable the searching of non-textual data by transforming it into text format, which can then be indexed and queried using off-the-shelf text search engines. We have called this family of approaches as *Surrogate Text Representation* (STR) [6,2,3].

In this short paper, we propose *Vec2Doc*, an alternative approach that transforms dense vectors into sparse representations that can be efficiently indexed using inverted indices. This technique allows us to take advantage of the well-established performance of inverted indices while also benefiting from the power of deep learning-generated dense vectors. Our method is specifically designed for dense vectors and allows easy indexing and retrieval using standard search engines that employ inverted indices. It extends the Scalar Quantization (SQ) STR approach [2] with a simple but effective idea to expand the codebook used for indexing, leading to improved performance of the inverted index.

The remainder of this paper is structured as follows: Section 2 provides an overview of related work, Section 3 presents the methodology of Vec2Doc, Section 3.1 details the experimental setup and results, and Section 4 concludes the paper and outlines directions for future research.

2 Background and Related Work

Most of the approaches for approximate metric search rely on transforming data objects into a different space where the search can be performed more efficiently. This is especially relevant when the original space has a high intrinsic dimensionality, when the dataset being searched is large, or when the actual distance to compare two data objects is computationally expensive. Examples of such techniques include transforming metric objects into binary sketches [8,7], permutations [4,16], and other pivot-based representations [9,15].

In 2010, Gennaro et al. [6] introduced a technique that utilized the distances between data objects and a set of reference objects (pivots) to map the data into a textual representation. Their objective was to convert a global descriptor into a sequence of terms resembling a text document, which could be processed by a text search engine such as Lucene. To achieve this, the mapping should be designed to approximate the original distance function, ensuring that the distance between textual documents and queries reflects the original similarity

between data objects and data queries. The main advantage of encoding data objects as text was the ability to leverage off-the-shelf text retrieval engines for performing similarity searches. This approach was initially referred to as the Surrogate Text Representation (STR) approach. However, over the years, various techniques have been proposed to transform descriptors into textual documents, and the term STR has come to encompass the broader family of approaches of this nature [2,1,3]. Some of these techniques have been designed to work on general metric spaces, while others are specialized for vector spaces.

In this work, our focus is on STR techniques specialized to index and search dense real vectors, such as data descriptors extracted with deep neural networks. These techniques can be mathematically formalized as space transformations of the form $f : \mathbb{R}^d \rightarrow \mathbb{N}^m$, where each original vector $\boldsymbol{y}$ is mapped into an integer-valued vector $\overline{\boldsymbol{y}}$. The key idea is to interpret $\overline{\boldsymbol{y}}$ as a term frequency vector based on a synthetic codebook $\mathcal{C} = \{\tau_1, \ldots, \tau_m\}$ of m terms. Consequently, the associated text document for vector $\boldsymbol{y}$ is obtained by concatenating the codebook terms with space separators, where each term τ_i is repeated a number of times equal to $\overline{y}_i$. We indicate with $T_{f,\mathcal{C}}(\cdot)$ the overall transformation from the original vectors to the text documents, which depends on both the function f and the used codebook $\mathcal{C}$. For example, if $f(\boldsymbol{y}) = \overline{\boldsymbol{y}} = [3, 1, 0, 2]$ and $\mathcal{C} =$ {"A", "B", "C", "D"}, the resulting text document associated with $\boldsymbol{y}$ would be $T_{f,\mathcal{C}}(\boldsymbol{y})$ ="A A A B D D". The rationale behind this approach is that the abstract transformation f corresponds to the function that precisely generates the vectors used internally by the search engine based on the *vector space model* [12], particularly in the case of a simple TF-weighting scheme.

To ensure compatibility with text retrieval engines and the efficiency of the inverted index, it is important that f generates a sparse vector with non-negative components. Various STR approaches exist, differing in their specific methods for handling negative values, achieving sparsification, and performing the final real-to-integer discretization. For example, in [2,3], the use of the Concatenated Rectified Linear Unit (CReLU) activation function is employed to prevent the presence of negative values in the transformed vectors. In [3] a Voronoi partitioning scheme is employed, where different codebooks are utilized for each partition. This approach aims to increase the sparsity of the transformed data, leading to more efficient indexing and retrieval processes.

3 Vec2Doc

The Vec2Doc is a generalization of the Scalar Quantization STR approach [2]. The SQ transforms a dense vector $\boldsymbol{y} \in \mathbb{R}^d$ into a term frequency vector $\overline{\boldsymbol{y}} \in \mathbb{N}^m$ through four main steps:

1. *Centering and Random Orthogonal Projection*: $\boldsymbol{y}_1 = R(\boldsymbol{y} - \boldsymbol{\mu}) \in \mathbb{R}^d$ where $R \in \mathbb{R}^{d \times d}$ is a random orthogonal matrix and $\boldsymbol{\mu} \in \mathbb{R}^d$ is set to center the data to zero mean. This step is used to uniform the distribution of vector components. In particular, the random rotation helps distribute the information along all the components of the vector in order to limit the

presence of unbalanced posting lists in the final inverted file (important for the efficiency of inverted indices). In [2] the centering operation is applied only to the data object but not to the queries in order to preserve dot-product similarities.

2. *Positivization*: $\boldsymbol{y}_2 = \text{CReLU}(\boldsymbol{y}_1) \in \mathbb{R}^{2d}$, where the Concatenated Rectified Linear Unit transformation [13] is applied to transform the vector into a positive one. The CReLU operation involves concatenating both the original vector and its negation and then setting all negative values to zero, i.e., $\text{CReLU}(\boldsymbol{v}) = \max([\boldsymbol{v}, -\boldsymbol{v}], \boldsymbol{0})$ where the max is applied element-wise.
3. *Sparsification*: $\boldsymbol{y}_3 = g_\gamma(\boldsymbol{y}_2) \in \mathbb{R}^{2d}$ where g_γ is a component-wise thresholding function, i.e., $g_\gamma(x) = x$ if $x > \gamma$, 0 otherwise.
4. *Integer Quantization*: $\boldsymbol{y}_4 = \lfloor s\boldsymbol{y}_3 \rfloor \in \mathbb{N}^{2d}$ where $\lfloor \cdot \rfloor$ denotes the floor function and s is a multiplication factor > 1 that works as a *quantization factor* to transform float components into integer.

The primary drawback of this approach is its limitation in terms of the dimensionality of the resulting term frequency vectors, which is fixed at $2d$, where d represents the dimensionality of the original vector. Consequently, the vocabulary size necessary for indexing the data with inverted files remains constant, posing an inconvenience when dealing with large datasets. To clarify further, if the number of posting lists is fixed, the length of each posting list may become excessive for large datasets, ultimately impacting search efficiency negatively.

To overcome this issue, in [3] we proposed the VP-SQ approach where the data is clustered in Voronoi cells and a different vocabulary can be used for each cell. In this paper, instead, we propose a simple but effective idea to extend the vocabulary size without using centroids. Nevertheless, our new Vec2Doc approach can be used alone or in combination with the Voronoi partitioning strategy to improve performance further.

The Vec2Doc transformation employs a semi-orthogonal transformation to expand the dimensionality of the vectors before the real-to-integer discretization process. Specifically, the initial step of the SQ approach (step 1. described above) is replaced with the following transformation:

$$\boldsymbol{y}_1 = A\boldsymbol{y} \in \mathbb{R}^m, \tag{1}$$

where $A \in \mathbb{R}^{m \times d}$ is a semi-orthogonal matrix (i.e., $A^T A = I$) with $m > d$. This transformation is applied to both data and query vectors without centering the former. The purpose of this transformation is to increase the vector dimensionality while preserving the dot product:

$$< A\boldsymbol{v}, A\boldsymbol{w} >= \boldsymbol{v}^T A^T A\boldsymbol{w} = \boldsymbol{v}^T \boldsymbol{w} =< \boldsymbol{v}, \boldsymbol{w} > .$$

Moreover, since the semi-orthogonal matrix is randomly chosen, it behaves similarly to a random rotation, effectively distributing information across the different dimensional components.

To sample a random semi-orthogonal matrix, we follow [11] and apply the following recurrent formula to a random normally-distributed real-valued matrix

$$A \leftarrow A - \frac{1}{2} A \left(A^T A - I \right) \tag{2}$$

that efficiently converge into a semi-orthogonal matrix in a few iterations.

3.1 Experiments

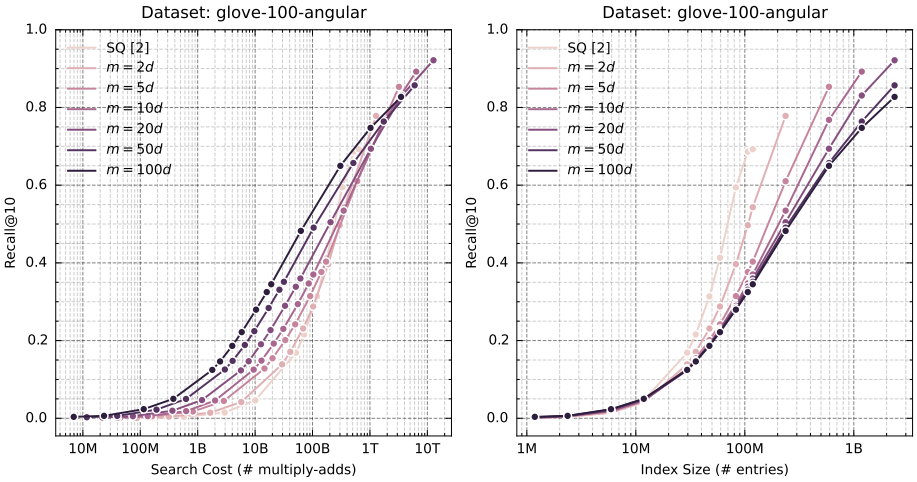
For evaluation, we adopt two approximate nearest neighbor benchmarks, which are Glove-100 [10] and NYTimes-256 [5], collecting $\sim$ 1M 100-dimensional and $\sim$ 280k 256-dimensional real-valued vectors as search set respectively. Both provide 10k test queries and desired results (100 nearest neighbors). We L_2-normalize all vectors to perform cosine similarities as inner products.

We measure the search effectiveness with the Recall@10 metric. For efficiency, we measure the search cost in an implementation-independent way by counting the number of accessed posts in the posting lists in the inverted index, which also corresponds to the number of multiply-add operations needed to compute scores for all data points. In addition, we measure the index size as the total number of posts in the posting lists.

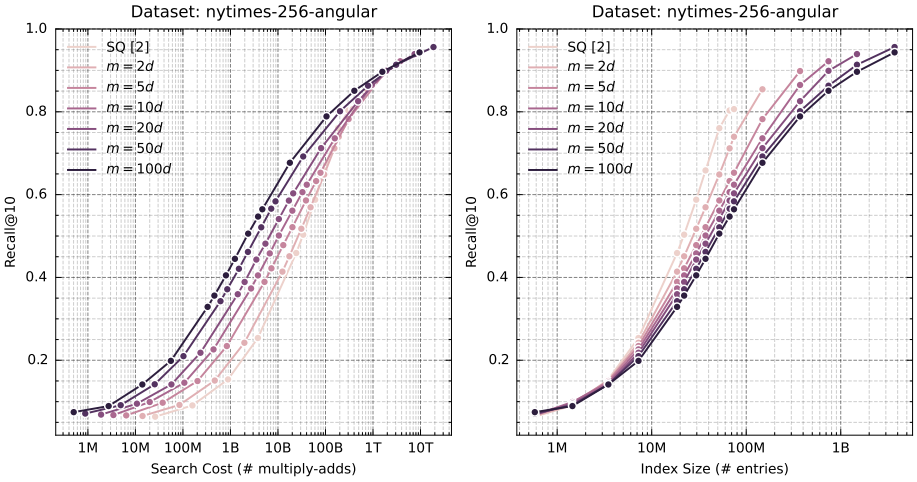
First, we compare our improved SQ scheme against the original one proposed in [2] in Figure 1. Each line is obtained by choosing the desired vocabulary size (i.e., the number of rows m of the semi-orthogonal transformation A in Equation 1) and varying the threshold γ that controls the sparsification level. We set the quantization factor $s = 10^5$ for all experiments. We note that, as m increases, the obtained recall increases when considering a fixed search cost, thus achieving a better recall-speed trade-off on both benchmarks. However, this is paid in terms of space; as m increases, the index size increases to maintain the same recall values. We also observe diminishing returns as m increases.

Next, we test our proposal in combination with the Voronoi partitioning scheme proposed in [3]. In brief, the search set is divided into c partitions via k-means at index time, and at query time, only the n partitions having the closest centroids to the query are accessed. In each partition, whichever STR technique can be adopted as long as each partition has its vocabulary that does not share tokens with other partitions. In Figure 2, we compare the Voronoi-partitioned version of our proposal with the Voronoi-partitioned SQ method (VP-SQ). Each line is obtained by varying all the method parameters (number of centroids $c \in \{128, 256, 512, 1024, 2048, 4096, 8192\}$, number of partitions accessed at query time $n \in \{1, 2, 4, 8, \ldots, c\}$, plus the parameters of the underline STR technique, i.e., sparsification threshold γ, vocabulary size m) and by keeping only the configurations belonging to the Pareto frontier. Due to the large number of parameter configurations to be tested, we report results only on the smaller NYTimes-256 benchmark. We can see that our proposal provides an improved effectiveness-efficiency trade-off also in the Voronoi-partitioning version.

These preliminary experiments demonstrate that the Vec2Doc approach can be effectively utilized in conjunction with Voronoi-partitioned STRs, offering an easy-to-implement strategy to enhance the vocabulary size within each Voronoi cell. In most cases, this achieves the optimal trade-off between efficiency and effectiveness.



(a) Results on Glove 100.



(b) Results on NYTimes 256.

Fig. 1: Effectiveness (Recall@K) vs. Search Cost (# of multiply-add needed to compute scores) and Index Size (# of entries in the index) of our proposal and of the SQ baseline [2]. Each line is obtained by choosing the desired vocabulary size m and varying the sparsification threshold γ.

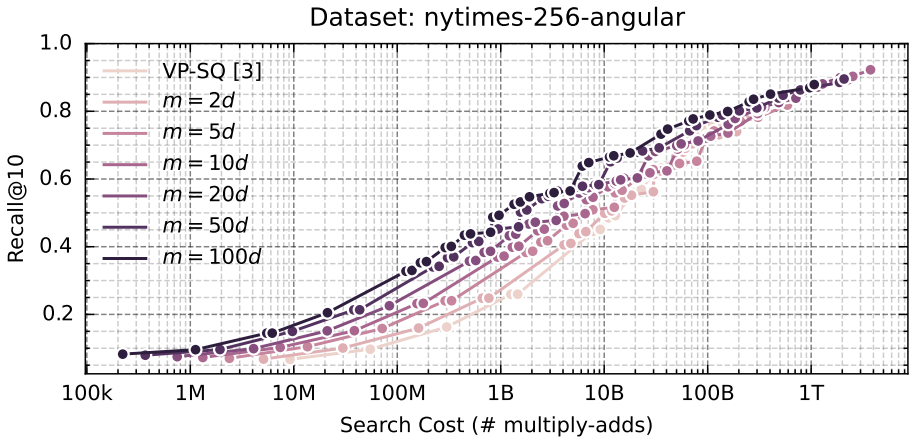


Fig. 2: NYTimes-256. Effectiveness (Recall@K) vs. Search Cost (# of multiply-add needed to compute scores) of the Voronoi-partitioned versions of our proposal and of SQ [3]. Each line is obtained by plotting the Pareto-optimal configurations when varying the number of partitions c, the number of accessed partitions n, the vocabulary size m, and the sparsification threshold γ.

4 Conclusion

In this paper, we have presented Vec2Doc, a novel approach for transforming dense vectors into sparse representations specifically designed to address the challenges of large-scale information retrieval tasks. Our approach enables an expansion of the vocabulary size utilized in STR encoding, thereby positively impacting search efficiency.

However, our approach is not without limitations. One of the challenges we aim to address in future work is the problem of out-of-distribution queries such as those arising from cross-modal features. Currently, Vec2Doc's performance may be hindered when dealing with cross-modal embeddings due to the application of the CReLU activation function. This limitation can lead to poor performance when handling datasets with mixed data modalities, such as text and images.

To overcome this challenge, future research will explore alternative feature transformation techniques that better handle cross-modal features. Additionally, we plan to investigate the scalability of Vec2Doc for increasingly larger datasets and the potential integration with other advanced IR systems to improve its applicability and performance further.

Acknowledgements This work was partially funded by AI4Media - A European Excellence Centre for Media, Society, and Democracy (EC, H2020 n. 951911), SUN - Social and hUman ceNtered XR (EC, Horizon Europe n. 101092612), and National Centre for HPC, Big Data and Quantum Computing – HPC (CUP B93C22000620006).

References

1. Amato, G., Bolettieri, P., Carrara, F., Falchi, F., Gennaro, C.: Large-scale image retrieval with elasticsearch. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. pp. 925–928 (2018)
2. Amato, G., Carrara, F., Falchi, F., Gennaro, C., Vadicamo, L.: Large-scale instance-level image retrieval. Inf. Process. Manage. **57**(6), 102100 (2020)
3. Carrara, F., Vadicamo, L., Gennaro, C., Amato, G.: Approximate nearest neighbor search on standard search engines. In: Similarity Search and Applications: 15th International Conference, SISAP 2022, Bologna, Italy, October 5–7, 2022, Proceedings. pp. 214–221. Springer (2022)
4. Chávez, E., Figueroa, K., Navarro, G.: Effective proximity retrieval by ordering permutations. IEEE Transactions on Pattern Analysis and Machine Intelligence **30**(9), 1647–1658 (2008)
5. Dua, D., Graff, C.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
6. Gennaro, C., Amato, G., Bolettieri, P., Savino, P.: An approach to content-based image retrieval based on the lucene search engine library. In: International Conference on Theory and Practice of Digital Libraries. pp. 55–66. Springer (2010)
7. Higuchi, N., Imamura, Y., Mic, V., Shinohara, T., Hirata, K., Kuboyama, T.: Nearest-neighbor search from large datasets using narrow sketches. In: ICPRAM. pp. 401–410 (2022)
8. Mic, V., Novak, D., Zezula, P.: Binary sketches for secondary filtering. ACM Transactions on Information Systems (TOIS) **37**(1), 1–28 (2018)
9. Novak, D., Zezula, P.: Ppp-codes for large-scale similarity searching. Transactions on Large-Scale Data-and Knowledge-Centered Systems XXIV: Special Issue on Database-and Expert-Systems Applications pp. 61–87 (2016)
10. Pennington, J., Socher, R., Manning, C.D.: GloVe: global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014)
11. Povey, D., Cheng, G., Wang, Y., Li, K., Xu, H., Yarmohammadi, M., Khudanpur, S.: Semi-orthogonal low-rank matrix factorization for deep neural networks. In: Interspeech. pp. 3743–3747 (2018)
12. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill, Inc., New York, NY, USA (1986)
13. Shang, W., Sohn, K., Almeida, D., Lee, H.: Understanding and improving convolutional neural networks via concatenated rectified linear units. In: Proceedings of the 33rd International Conference on Machine Learning. ICML 2016, vol. 48, pp. 2217–2225. JMLR.org (2016)
14. Simhadri, H.V., Williams, G., Aumüller, M., Douze, M., Babenko, A., Baranchuk, D., Chen, Q., Hosseini, L., Krishnaswamny, R., Srinivasa, G., et al.: Results of the neurips'21 challenge on billion-scale approximate nearest neighbor search. In: NeurIPS 2021 Competitions and Demonstrations Track. pp. 177–189. PMLR (2022)
15. Vadicamo, L., Connor, R., Chávez, E.: Query filtering using two-dimensional local embeddings. Information Systems **101**, 101808 (2021)
16. Vadicamo, L., Gennaro, C., Falchi, F., Chávez, E., Connor, R., Amato, G.: Re-ranking via local embeddings: a use case with permutation-based indexing and the nsimplex projection. Information Systems **95**, 101506 (2021)